IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
Before the Board of Patent Appeals and Interferences

In re Patent Application of

EVANS et al

Serial No. 10/601,575

Filed: June 24, 2003

Atty Dkt. JRL-550-445
   C#   M#
Confirmation No. 8224

TC/A.U.: 2181

Examiner: Lee, Chun Kuan

Date: June 20, 2008

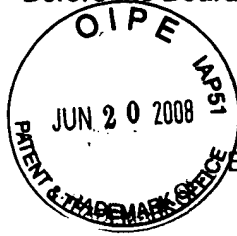*[Stamp: O.I.P.E. JUN 20 2008 PATENT & TRADEMARK OFFICE]*

Title: SYNCHRONIZATION BETWEEN PIPELINES IN A DATA PROCESSING APPARATUS UTILIZING A SYNCHRONIZATION QUEUE

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

☐ **Correspondence Address Indication Form Attached.**

☐ <u>NOTICE OF APPEAL</u>
Applicant hereby **appeals** to the Board of Patent Appeals and Interferences
from the last decision of the Examiner twice/finally rejecting $510.00 (1401)/$255.00 (2401) $
applicant's claim(s).

☒ An appeal **BRIEF** is attached in the pending appeal of the
above-identified application $510.00 (1402)/$255.00 (2402) $ 510.00

☐ Credit for fees paid in prior appeal without decision on merits   -$ (   )

☐ A reply brief is attached.   **(no fee)**

☐ Petition is hereby made to extend the current due date so as to cover the filing date of this
paper and attachment(s)   One Month Extension $120.00 (1251)/$60.00 (2251)
                                       Two Month Extensions $460.00 (1252)/$230.00 (2252)
                                       Three Month Extensions $1050.00 (1253/$525.00 (2253)
                                       Four Month Extensions $1640.00 (1254/$820.00 (2254) $

☐ "Small entity" statement attached.

Less   month extension previously paid on   -$(   )

TOTAL FEE ENCLOSED $ 510.00

☒ **CREDIT CARD PAYMENT FORM ATTACHED.**

Any future submission requiring an extension of time is hereby stated to include a petition for such time extension.
The Commissioner is hereby authorized to charge any <u>deficiency</u>, or credit any overpayment, in the fee(s) filed, or
asserted to be filed, or which should have been filed herewith (or with any paper hereafter filed in this application by this
firm) to our **Account No. 14-1140**. A <u>duplicate</u> copy of this sheet is attached.

901 North Glebe Road, 11th Floor
Arlington, Virginia 22203-1808
Telephone: (703) 816-4000
Facsimile: (703) 816-4100
JRL:maa

NIXON & VANDERHYE P.C.
By Atty: John R. Lastova, Reg. No. 33,149

Signature: _____

1351429

COPY

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
Before the Board of Patent Appeals and Interferences

In re Patent Application of

EVANS et al

Serial No. 10/601,575

Filed: June 24, 2003

Atty Dkt. JRL-550-445

    C#   M#

Confirmation No. 8224

TC/A.U.: 2181

Examiner: Lee, Chun Kuan

Date: June 20, 2008

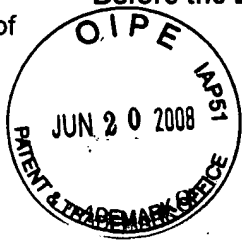*(stamp: O.I.P.E. JUN 20 2008 PATENT & TRADEMARK OFFICE)*

Title: SYNCHRONIZATION BETWEEN PIPELINES IN A DATA PROCESSING APPARATUS UTILIZING A SYNCHRONIZATION QUEUE

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

☐ **Correspondence Address Indication Form Attached.**

☐ **NOTICE OF APPEAL**
Applicant hereby **appeals** to the Board of Patent Appeals and Interferences from the last decision of the Examiner twice/finally rejecting $510.00 (1401)/$255.00 (2401) $ applicant's claim(s).

☒ An appeal **BRIEF** is attached in the pending appeal of the above-identified application $510.00 (1402)/$255.00 (2402) $ 510.00

☐ Credit for fees paid in prior appeal without decision on merits   -$ (   )

☐ A reply brief is attached.   **(no fee)**

☐ Petition is hereby made to extend the current due date so as to cover the filing date of this paper and attachment(s)
    One Month Extension $120.00 (1251)/$60.00 (2251)
    Two Month Extensions $460.00 (1252)/$230.00 (2252)
    Three Month Extensions $1050.00 (1253/$525.00 (2253)
    Four Month Extensions $1640.00 (1254/$820.00 (2254) $

☐ "Small entity" statement attached.

Less   month extension previously paid on   -$(   )

TOTAL FEE ENCLOSED $ 510.00

☒ CREDIT CARD PAYMENT FORM ATTACHED.

Any future submission requiring an extension of time is hereby stated to include a petition for such time extension. The Commissioner is hereby authorized to charge any <u>deficiency</u>, or credit any overpayment, in the fee(s) filed, or asserted to be filed, or which should have been filed herewith (or with any paper hereafter filed in this application by this firm) to our **Account No. 14-1140.** A <u>duplicate</u> copy of this sheet is attached.

901 North Glebe Road, 11th Floor
Arlington, Virginia 22203-1808
Telephone: (703) 816-4000
Facsimile: (703) 816-4100
JRL:maa

NIXON & VANDERHYE P.C.
By Atty: John R. Lastova, Reg. No. 33,149

Signature: _____

1351429

In re Patent Application of

EVANS et al

Serial No. 10/601,575

Filed: June 24, 2003

Atty. Ref.: 550-445

Group: 2181

Examiner: Lee, Chun Kuan

For: SYNCHRONISATION BETWEEN PIPELINES IN A DATA PROCESSING APPARATUS UTILIZING A SYNCHRONISATION QUEUE

Before the Board of Patent Appeals and Interferences

BRIEF FOR APPELLANT
On Appeal From Final Rejection
From Group Art Unit 2181

John R. Lastova
NIXON & VANDERHYE P.C.
11th Floor, 901 North Glebe Road
Arlington, Virginia 22203-1808
(703) 816-4025
Attorney for Appellants
EVANS et al. and
ARM Limited

1

1346946

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Patent Application of

EVANS et al   Atty. Ref.: 550-445

Serial No. 10/601,575   Group: 2181

Filed: June 24, 2003   Examiner: Lee, Chun Kuan

For: SYNCHRONISATION BETWEEN PIPELINES IN A DATA
        PROCESSING APPARATUS UTILIZING A SYNCHRONISATION
        QUEUE

* *

June 20, 2008

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF

I. REAL PARTY IN INTEREST

The real party in interest is the assignee, ARM Limited, a United Kingdom

corporation.

II. RELATED APPEALS AND INTERFERENCES

There are no other appeals related to this subject application. There are no

interferences related to this subject application.

1346946

III. STATUS OF CLAIMS

Claims 1-22, 24, 25, 27-39, 41, 42, 44, and 45 are pending and rejected.

Claims 1-22, 24, 25, 27-39, 41, 42, 44, and 45 are appealed.

IV. STATUS OF AMENDMENTS

An amendment was filed after final on January 15, 2008 and has been entered for purposes of appeal as indicated in the advisory action dated February 13, 2008. That amendment overcomes the objections to the specification, abstract, and claims regarding the spelling of "synchronization," as well as the rejections under 35 U.S.C. §112, second paragraph of claims 24 and 41 relating to claim dependency based on a canceled claim.

V. SUMMARY OF THE CLAIMED SUBJECT MATTER

The technology in this case relates to synchronizing pipelines in a data processing apparatus. A main pipelined processor 40 works together with a pipelined coprocessor 110 as shown in the example of Figure 1. Typically, each coprocessor instruction is arranged to be routed through both of these pipelines with the intent that the coprocessor is to run more or less in step with the main processor. Thus, synchronization is needed for interaction between the various pipeline stages of the main processor and the coprocessor during execution of a coprocessor instruction. For example, coprocessor instructions may be cancelled by the main processor if a condition code specified by the coprocessor instruction is not met, or

2

the entire coprocessor pipeline may need to be flushed in the event of a mispredicted branch that has resulted in the coprocessor instruction being executed. Further, data may need to be passed between the main processor and the coprocessor in the event that the coprocessor instructions define load or store operations.

Up to now, a coprocessor pipeline has been synchronized with the main processor pipeline using a "tightly-coupled scheme" in which signals with fixed timing are passed from one pipeline to the other. These signals mainly cause "stalls" in one pipeline when the other pipeline stalls in order to maintain synchronisation. However, there are other complicating factors, for example, when the main pipeline needs to cancel the coprocessor instruction, or when the pipelines need to be flushed, which significantly complicate the stall interactions between the main processor and the coprocessor. Longer pipelines make it more difficult to achieve synchronization between pipelines using this "tightly-coupled scheme."

A major constraint on the coprocessor interface is that it must operate over a two cycle delay. Any signal passing from the main processor to the coprocessor, or vice versa, must be given a whole clock cycle to propagate from one to the other, and as a result, cannot be processed until the following clock cycle. Thus, a signal crossing the interface must be clocked out of a register on one side of the interface and clocked directly into another register on the other side, and no combinatorial process must intervene. This constraint arises from the fact that the main processor (also referred to in the specification as the "processor core") and the coprocessor

3

may be placed a considerable distance apart and generous timing margins must be

allowed for to cover signal propagation times. This is particularly true in situations

where the coprocessor may be designed separately from the main processor, for

example, by a different party. This delay in signal propagation makes it difficult to

maintain pipeline synchronization using the "tightly-coupled" synchronization

technique.

To solve this problem, at least one synchronizing queue couples a

predetermined pipeline stage in one of the pipelines with a partner pipeline stage in

the other of the pipelines. The predetermined pipeline stage causes a token to be

placed in the synchronizing queue when processing a coprocessor instruction. The

partner pipeline stage then processes that coprocessor instruction upon receipt of the

token from the synchronizing queue, thereby synchronizing the first and second

pipelines at that point without passing signals with fixed timing between the

pipelines. The token includes a tag which uniquely identifies the coprocessor

instruction to which the token relates.

This token-based pipeline synchronization technique allows some "slack"

between the two pipelines so that strict synchronization at all stages is not necessary.

At the same time, the pipelines are correctly synchronized for crucial transfers of

information. The claimed approach can be viewed as a data-driven, loosely-coupled,

synchronization scheme, in contrast to the control-driven, tightly-coupled scheme

where signals with fixed timing are passed between the pipelines.

The following claim charts provide a mapping of the independent claims onto non-limiting example text from the specification and figures by reference numerals where appropriate. This mapping is not intended to be used for claim construction.

1. A data processing apparatus, comprising:	Figure 1.
a main processor configured to execute a sequence of instructions, the main processor comprising a first pipeline having a first plurality of pipeline stages;	Processor 40 in Figure 1. Pipeline 30 in Figure 1 and stages 190-290 in Figure 2A.
a coprocessor configured to execute coprocessor instructions in said sequence of instructions, the coprocessor comprising a second pipeline having a second plurality of pipeline stages, and each coprocessor instruction being arranged to be routed through both the first pipeline and the second pipeline; and	Coprocessor 110 in Figure 1. Page 16, lines 15-21. Second pipeline 130 in Figure 1. Stage 205-275 in Figure 2B. Page 16, lines 22-30.
at least one synchronizing queue including a first-in-first-out (FIFO) buffer having a predetermined plurality of entries and coupling a predetermined pipeline stage in one of the pipelines with a partner pipeline stage in the other of the pipelines, the predetermined pipeline stage being configured to cause a token to be placed in an entry of the synchronizing queue when	Queues 285 and 295 in Figures 2A and 2B, page 18, lines 20-26, and elements 300-410 in Figures 3 and 4. See page 22, line 11- page 24, line 7, page 25, lines 4-19, and page 11, lines 11-16.

5

processing a coprocessor instruction, the token including a tag which uniquely identifies the coprocessor instruction to which the token relates, and the partner pipeline stage being configured to process that coprocessor instruction upon receipt of the token from the synchronizing queue, thereby synchronizing the first and second pipelines between the predetermined pipeline stage and the partner pipeline stage without passing signals with fixed timing between the pipelines.	See page 29, lines 10-29 and page 10, lines 13-26. See example in Figures 10-15 and pages 30-35. Page 3, lines 26-29.

29. A method of synchronization between pipelines in a data processing apparatus, the data processing apparatus comprising a main processor configured to execute a sequence of instructions and a coprocessor configured to execute coprocessor instructions in said sequence of instructions, the main processor comprising a first pipeline having a first plurality of pipeline stages, and the coprocessor comprising a second pipeline having a second plurality of pipeline stages, and each coprocessor instruction being arranged to be routed through both the first pipeline and the second pipeline, the method comprising the	Processor 40 in Figure 1. Pipeline 30 in Figure 1 and stages 190-290 in Figure 2A.. Coprocessor 110 in Figure 1. Page 16, lines 15-21. Second pipeline 130 in Figure 1. Stage 205-275 in Figure 2B. Page 16, lines 22-30.

6

steps of:	
(a) coupling a predetermined pipeline stage in one of the pipelines with a partner pipeline stage in the other of the pipelines via a synchronizing queue including a first-in-first-out (FIFO) buffer having a predetermined plurality of entries;	Queues 285 and 295 in Figures 2A and 2B, page 18, lines 20-26, and elements 300-410 in Figures 3 and 4. See page 22, line 11- page 24, line 7, page 25, lines 4-19, and page 11, lines 11-16.
(b) placing a token in an entry of the synchronizing queue when the predetermined pipeline stage is processing a coprocessor instruction, the token including a tag which uniquely identifies the coprocessor instruction to which the token relates;	See page 29, lines 10-29 and page 10, lines 13-26.
(c) upon receipt of the token from the synchronizing queue by the partner pipeline stage, processing the coprocessor instruction within the partner pipeline stage;	See example in Figures 10-15 and pages 30-35.
wherein synchronization of the first and second pipelines between the predetermined pipeline stage and the partner pipeline stage is obtained without passing signals with fixed timing between the pipelines.	Page 3, lines 26-29.

7

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The question to be decided by the Board is whether the rejection of claims

1-22, 24, 25, 27-39, 41, 42, 44, and 45 under 35 U.S.C. §103 based on Gearty

(6,477,638) and Martin (6,381,692) is improper.

VII. ARGUMENT

The Rejection Under 35 U.S.C. §103 Based On Gearty and Martin Is Improper

1.   Gearty Lacks Many of the Claimed Features

Gearty describes a *tightly-coupled* system where a coprocessor pipeline is

synchronized with the main processor pipeline by passing signals with *fixed timing*

from one pipeline to the other. A problem with tightly-coupled schemes is that as

the length of pipeline processors increases, it is more difficult to maintain pipeline

synchronization because of signal propagation delays that make it difficult to

ensure signals are passed between the pipelines with the required fixed timing.

Independent claims 1 and 29 solve this problem using a token-based pipeline

synchronization technique where at least one synchronizing queue couples a

predetermined pipeline stage in one of the pipelines with a partner pipeline stage

in the other of the pipelines. The predetermined pipeline stage and the partner

pipeline stage transfer tokens to achieve a *loosely-coupled* synchronization

scheme.

As admitted by the Examiner, Gearty's tightly-coupled scheme lacks at least *three* features recited in the independent claims 1 and 29. The *first* is a synchronizing queue that includes a FIFO buffer with a predetermined plurality of entries. The presence of multiple entries in the queue results in variable timing in the transfer of tokens between the predetermined pipeline stage and the partner pipeline stage. If at the time a token is placed in the queue, there are no further entries ahead of the token in the queue, then that token is transferred more quickly than if at the time the token is placed in the queue there are multiple tokens ahead of that token in the queue. The *second* missing feature is that the token includes a tag which uniquely identifies the coprocessor instruction to which the token relates. Because there is no fixed timing for the transfer of information from the predetermined pipeline stage to the partner pipeline stage, the token uniquely identifies the coprocessor instruction to which the token relates so that the partner pipeline stage can react appropriately. The *third* missing feature is that synchronization is achieved "without passing signals with fixed timing between the pipelines."

Although the Examiner turns to Martin in the hope of remedying the many deficiencies in the primary Gearty reference, Martin is unavailing. As a result, their combination fails to teach the combination of features in claims 1 and 29. Nor does either reference achieve the flexibility and reliability achieved by the claimed loosely-coupled synchronization scheme. Indeed, the claimed loosely-

coupled synchronization scheme can be used in situations where Gearty's tightly-coupled synchronization scheme may well not even work, such as in systems where the length of the pipelined processors is large, where signal delay propagation makes it difficult to use a tightly coupled synchronization scheme, etc. Although the amount of slip between the pipelines is allowed to vary with the claimed loosely-coupled synchronization scheme, the pipelines are still properly synchronized for crucial transfers of information.

2.   <u>Gearty And Martin Lack The Claimed Synchronizing Queue FIFO Buffer</u>

Claims 1 and 29 recite a main processor with a first pipeline, a coprocessor with a second pipeline, and a synchronizing queue coupling a predetermined pipeline stage in one of the pipelines with a partner pipeline stage in the other of the pipelines. This synchronizing queue which includes a FIFO buffer having a predetermined plurality of entries is missing from Gearty, and as explained below, is also missing from Martin.

Even though Martin can be coupled to a coprocessor, Martin only discusses the structure of the main MIPS R3000 RISC processor (see column 3, lines 10-13), which is an <u>asynchronous</u> processor, in contrast to the synchronous design recited in claims 1 and 29 and described in Gearty. Indeed, most of the Examiner's references are to Figure 1 which illustrates the main asynchronous processor. Column 4, lines 22-24 states: "FIG. 1 is a block diagram illustrating one embodiment 100 of an *asynchronous* pipeline architecture implementing the

MIPS R3000." The FIFO structures 160a and 160b shown in Figure 1 are inside

the main *asynchronous* processor 100. In contrast, claims 1 and 29 are directed to

synchronizing pipelined processors.

Martin's FIFO 160a is coupled between the decoder and the write back unit

to store and transfer ordering information to the write back unit (see column 2,

lines 34-36), and the FIFO queue 160b is disposed between the program counter

unit and the write back unit to store and transfer a program counter signal to the

write back unit (see column 2, lines 39-41). Neither FIFO couples a

predetermined pipeline stage in the main processor 100 with a partner pipeline

stage in a coprocessor or a predetermined pipeline stage in a coprocessor with a

partner pipeline stage in the main processor.

The text at column 9, lines14-29 referred to by the Examiner describes the

need for FIFO queues 160a and 160b which stems from the fact that Martin's

processor is *asynchronous*, i.e., a processor where each functional block of the

processor only operates when data arrives, rather than the various functional

blocks all being operated in accordance with a system clock signal. The FIFO

queues 160a and 160b have nothing to do with synchronization between a main

processor and a coprocessor.

Accordingly, nothing in Martin discloses a synchronizing queue (Martin is

directed to an asynchronous design) coupling a predetermined pipeline stage in

one of the pipelines (either the pipeline of the main processor or the coprocessor)

11

with a partner pipeline stage in the other of the pipelines (either the coprocessor or the main processor). The fact that the main processor 100 may be coupled to a coprocessor does not disclose or suggest this claimed feature and has no relevance to the arguments that the Examiner seeks to make based on the teaching of Martin.

There is no reason to include Martin's asynchronous system queues 160a and 160b in a synchronous system. In fact, a person of ordinary skill in the art would have been motivated not to include Martin's asynchronous system queues 160a and 160b into Gearty's tightly-coupled synchronized system because introducing Martin's asynchronous behavior into Gearty would have undermined the very objective that the Gearty's tightly-coupled synchronized system is trying to achieve.

The mere fact that a prior art reference could be modified does not mean that it is obvious to modify the prior art reference. See, for example, *In re Fulton*, 391 F.3d 1195, 1200 (Fed. Cir. 2004). The Federal Circuit has made clear that a proposed modification which renders a prior art reference inoperable for its intended purpose is inappropriate for an obviousness inquiry. *In re Gordon*, 733 F.2d 900, 902 (Fed. Cir. 1992).

3. Gearty And Martin Lack The Token Tag For A Coprocessor Instruction

Regarding the missing "tag which uniquely identifies the coprocessor instruction to which the token is related," the Examiner refers to column 6, lines 11-23 of Martin. This text describes fetching instructions from the instruction

cache by the fetch unit, and in particular, to the standard comparison of a tag

portion of a fetch address with the tag entries in the instruction cache to determine

whether the requested instruction is within the instruction cache, i.e., whether a hit

or miss condition exists.

This text from Martin lacks several features of what is claimed. First, the

fetched instruction at column 6, lines 11-23 is not a coprocessor instruction, as

claimed; rather, it is an instruction for the main processor 100. Second, the tag

values held in an instruction cache are merely a portion of an instruction address

and do not "uniquely" identify particular instances of an instruction, as claimed.

Third, and significantly, Martin's instruction cache tags have nothing to do with

placing a token in a synchronizing queue between a main processor and a

coprocessor. Indeed, Martin fails to describe any details of how the main

processor should be coupled to a coprocessor. Martin's instruction cache tags fail

to teach the claimed token including a *tag* which *uniquely identifies the

coprocessor instruction to which the token relates*. A lookup operation performed

within an instruction cache to determine whether an instruction is present in the

cache is entirely unrelated to placing a token in a synchronizing queue between a

processor and a coprocessor.

The Examiner constructs this portion of the rejection using an unreasonable

contention that because Martin teaches a tag in a completely different context and

for a completely different use, the Martin renders obvious the use of the claimed

tag within at token placed in an entry of a synchronizing queue. That simply is not the standard for obviousness. By this kind of reasoning, the teaching of a tag in one reference necessarily prohibits any patentable subject matter that involves a tag. The fallacy of this logic is even more evident if one substitutes "computer" for "tag" in the Examiner's reasoning and contends (absurdly) that the teaching of a computer in one reference necessarily prohibits any patentable subject matter that involves a computer.

4. Gearty and Martin Lack Synchronization Between The First And Second Pipelines Occurs Without Passing Signals With Fixed Timing Between The Pipelines

Regarding this third admitted missing feature from Gearty, the Examiner refers to column 1, lines 41-66 of Martin. Here, Martin provides general background information on asynchronous processors. But the present claims are directed to achieving synchronization between two separate processors: a main processor and a coprocessor used to execute certain coprocessor instructions appearing in the sequence of instructions executed by the main processor. This is clear from the language of claim 1 which states that by using at least one synchronizing queue including a FIFO buffer having a predetermined plurality of entries, and by placing tokens in that queue that include a tag which uniquely identifies the coprocessor instruction to which the token relates, the first and second pipelines (one being in the main processor and the other being in the coprocessor) are synchronized between the predetermined pipeline stage and the

14

partner pipeline stage without passing signals with fixed timing <u>between the</u>

<u>pipelines</u>. There is no teaching in Martin of passing any signals between pipelines

in different processors, with fixed timing or otherwise.

4.   <u>The Rationale For Modifying Gearty With Martin Is</u>
       <u>Unreasonable</u>

A prior art reference must be considered in its entirety, i.e., as a whole,

including portions that would lead away from the claimed invention. *W.L. Gore &*

Associates, Inc. v. Garlock, Inc., 721 F.2d 1540 (Fed. Cir. 1983), cert. denied, 469

U.S. 851 (1984). The final rejection states that it would be obvious to "include

<u>Martin</u>'s asynchronous processor's FIFO and tag into <u>Gearty</u>'s data processing

apparatus for the benefit of implementing asynchronous processor having simpler

architecture and faster processing speed." But that benefit only makes sense in

Martin alone. Martin wants asynchronous operation—not synchronized. But

Gearty's objective is synchronized operation. The modification the Examiner is

proposing undermines Gearty's synchronization objective which makes it

unreasonable. See *In re Gordon*. If one or both of Gearty's CPU and FPU each

incorporated Martin's FIFOs so that Gearty would be asynchronous as the

Examiner proposes, those FIFOs would not produce the claimed synchronization

between the first and second pipelines without passing signals. In other words, the

Examiner's modification to Gearty, in addition to not making technical sense, also

moves Gearty even further away from what is claimed. Moreover, in the modified

system, there would not be a synchronizing queue <u>between</u> the pipelines.

<div align="center">15</div>

5. Dependent Claim Features Are Not Taught

Given the many deficiencies noted above with respect to the independent

claims, it is not surprising that many of the dependent claim features are also

missing. For example, claims 2 and 30 identify that there are a plurality of the

synchronizing queues, with each queue coupling a predetermined pipeline stage in

one of the pipelines with a partner pipeline stage in the other of the pipelines.

Although Martins shows two FIFOs 160a and 160b, neither queue couples a

pipeline stage in the main processor with a pipeline stage in the coprocessor, or

vice versa, and indeed as explained above, both of the queues shown in Martin

actually reside within the main processor. The Examiner refers to Figure 4 of

Gearty. But as already conceded by the Examiner for claim 1, Gearty does not

disclose a synchronizing queue, as defined in claim 1, because the various latches

provided between the CPU pipeline and the FPU pipeline in Gearty cannot be

considered to be synchronizing queues "including a FIFO buffer having a

predetermined plurality of entries," as required by Claim 1.

As another example, claims 3 and 31 specify that one of the synchronizing

queues is an instruction queue and that the first pipeline (in the main processor)

places a token identifying the coprocessor instruction in the instruction queue, so

that the second pipeline (in the coprocessor) begins processing the relevant

coprocessor instruction upon receipt of the token. The Examiner refers column

11, lines 62 to 67 in Gearty which merely describes synchronizing instructions

between stages 170 and 126 shown in Figure 4, which are partway through the pipeline stages in both the main processor and the coprocessor. Hence, this section is describing a synchronization that occurs sometime after the coprocessor has began processing the coprocessor instruction. Column 13, lines 14 to 35 refers to a sequence of events that occurs when the signal on line 282 in Figure 7 is deactivated, which is towards the end of both pipeline stages. So again, this process takes place significantly after the coprocessor has begun processing the coprocessor instruction.

Claims 4 is one example of the arrangement in claim 3 specifying that the instruction queue exists between the fetch stage in the first pipeline (of the main processor) and the decode stage in the second pipeline (of the coprocessor). The Examiner argues that this feature is shown in Figure 4 of Gearty. But as is apparent from Figure 3 of Gearty, (see also column 6, lines 18 to 20), the fetch stage 164 is shared between both the main processor and the coprocessor, and that stage is not even shown in Figure 4. Presumably, the Examiner equates the decode stage in the second pipeline with the floating point decode pipeline stage 178 shown towards the upper right hand side of Figure 4. However, it is clear that there is no queue coupling the fetch stage of the main processor (not shown in Figure 4) with the decode stage in the coprocessor.

Given these deficiencies with respect to claim 4, the features of claim 5 are also missing from the Gearty/Martin combination. Claim 5 provides more details

17

as to how the fetch stage in the first pipeline (the main processor) places tokens in the instruction queue. Again, the Examiner's statement that the fetch stage of the first pipeline and the decode stage of the second pipeline are shown as connected in Figure 4 is incorrect because the fetch stage is not even shown in Figure 4, and in any case, is shared between the two pipelines as described earlier.
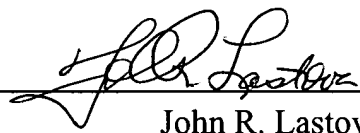
Many other dependent claim features missing from Gearty and Martin could be argued. But those additional distinctions are not believed necessary given the many distinctions already established.

VIII. CONCLUSION

Any one of the above clear errors defeats the rejection based on Gearty and Martin. The final rejection under 35 U.S.C. §103 should be reversed, and the application passed to allowance.

Respectfully submitted,

NIXON & VANDERHYE P.C.

By: _____

John R. Lastova

JRL/maa
Appendix A - Claims on Appeal

Reg. No. 33,149

18

1346946

IX. CLAIMS APPENDIX

1.   (previously presented) A data processing apparatus, comprising:

a main processor configured to execute a sequence of instructions, the main

processor comprising a first pipeline having a first plurality of pipeline stages;

a coprocessor configured to execute coprocessor instructions in said sequence of

instructions, the coprocessor comprising a second pipeline having a second plurality of

pipeline stages, and each coprocessor instruction being arranged to be routed through

both the first pipeline and the second pipeline; and

at least one synchronizing queue including a first-in-first-out (FIFO) buffer having

a predetermined plurality of entries and coupling a predetermined pipeline stage in one of

the pipelines with a partner pipeline stage in the other of the pipelines, the predetermined

pipeline stage being configured to cause a token to be placed in an entry of the

synchronizing queue when processing a coprocessor instruction, the token including a tag

which uniquely identifies the coprocessor instruction to which the token relates, and the

partner pipeline stage being configured to process that coprocessor instruction upon

receipt of the token from the synchronizing queue, thereby synchronizing the first and

second pipelines between the predetermined pipeline stage and the partner pipeline stage

without passing signals with fixed timing between the pipelines.

2. (previously presented)   A data processing apparatus as claimed in Claim 1,

further comprising a plurality of said synchronizing queues, each said synchronizing

queue coupling a predetermined pipeline stage in one of the pipelines with a partner pipeline stage in the other of the pipelines.

3. (previously presented)   A data processing apparatus as claimed in Claim 1, wherein one of the at least one synchronizing queues is an instruction queue, the predetermined pipeline stage is in the first pipeline and is arranged to cause a token identifying a coprocessor instruction to be placed in the instruction queue, and the partner pipeline stage is in the second pipeline and is configured upon receipt of the token to begin processing the coprocessor instruction identified by the token.

4. (previously presented)   A data processing apparatus as claimed in Claim 3, wherein the predetermined pipeline stage is a fetch stage in the first pipeline and the partner pipeline stage is a decode stage in the second pipeline, that decode stage being configured to decode the coprocessor instruction upon receipt of the token.

5. (previously presented)   A data processing apparatus as claimed in Claim 4, wherein the fetch stage in the first pipeline is configured to cause a token to be placed in the instruction queue for each instruction in the sequence of instructions, and the decode stage in the second pipeline is arranged to decode each instruction upon receipt of the associated token in order to determine whether that instruction is a coprocessor instruction that requires further processing by the coprocessor.

6. (previously presented)   A data processing apparatus as claimed in Claim 1, wherein one of the at least one synchronizing queues is a cancel queue, the predetermined pipeline stage is in the first pipeline and is arranged to cause to be placed in the cancel queue a token identifying whether a coprocessor instruction at that predetermined pipeline stage is to be cancelled, and the partner pipeline stage is in the second pipeline and is configured upon receipt of the token from the cancel queue, and if the token identifies that the coprocessor instruction is to be cancelled, to cause that coprocessor instruction to be cancelled.

7. (original)   A data processing apparatus as claimed in Claim 6, wherein the predetermined pipeline stage is an issue stage in the first pipeline, and the partner pipeline stage is a stage following an issue stage in the second pipeline.

8. (previously presented)   A data processing apparatus as claimed in Claim 6, wherein the partner pipeline stage is configured upon receipt of the token from the cancel queue, and if the token identifies that the coprocessor instruction is to be cancelled, to remove the coprocessor instruction from the second pipeline.

9. (previously presented)   A data processing apparatus as claimed in Claim 1, wherein one of the at least one synchronizing queues is a finish queue, the predetermined pipeline stage is in the first pipeline and is arranged to cause to be placed in the finish queue a token identifying permission for a coprocessor instruction at that predetermined

-A3-

pipeline stage to be retired from the second pipeline, and the partner pipeline stage is in the second pipeline and is configured upon receipt of the token from the finish queue, and if the token identifies that the coprocessor instruction is permitted to be retired, to cause that coprocessor instruction to be retired.

10. (original) A data processing apparatus as claimed in Claim 9, wherein the predetermined pipeline stage is a write back stage in the first pipeline, and the partner pipeline stage is a write back stage in the second pipeline.

11. (previously presented) A data processing apparatus as claimed in Claim 1, wherein one of the at least one synchronizing queues is a length queue, the predetermined pipeline stage is in the second pipeline and is arranged, for a vectored coprocessor instruction, to cause to be placed in the length queue a token identifying length information for the vectored coprocessor instruction, and the partner pipeline stage is in the first pipeline and is configured upon receipt of the token from the length queue to factor the length information into the further processing of the vectored coprocessor instruction within the first pipeline.

12. (original) A data processing apparatus as claimed in Claim 11, wherein the predetermined pipeline stage is a decode stage in the second pipeline, and the partner pipeline stage is a first execute stage in the first pipeline.

13. (previously presented) A data processing apparatus as claimed in Claim 1, wherein one of the at least one synchronizing queues is an accept queue, the predetermined pipeline stage is in the second pipeline and is arranged to cause to be placed in the accept queue a token identifying whether a coprocessor instruction in that predetermined pipeline stage is to be accepted for execution by the coprocessor, and the partner pipeline stage is in the first pipeline and is configured upon receipt of the token from the accept queue, and if the token identifies that the coprocessor instruction is not to be accepted, to cause that coprocessor instruction to be rejected by the main processor.

14. (original) A data processing apparatus as claimed in Claim 13, wherein the predetermined pipeline stage is an issue stage in the second pipeline, and the partner pipeline stage is a second execute stage in the first pipeline.

15. (previously presented) A data processing apparatus as claimed in Claim 14, wherein the partner pipeline stage is configured upon receipt of the token from the accept queue, and if the token identifies that the coprocessor instruction is not to be accepted, to remove the coprocessor instruction from the first pipeline.

16. (previously presented) A data processing apparatus as claimed in Claim 1, wherein one of the at least one synchronizing queues is a store queue used when the coprocessor instruction is a store instruction configured to cause data items to be transferred from the coprocessor to memory accessible by the main processor, the

-A5-

predetermined pipeline stage is in the second pipeline and is arranged, when processing

one of said store instructions, to cause to be placed in the store queue a token identifying

each data item to be transferred, and the partner pipeline stage is in the first pipeline and

is configured upon receipt of each token from the store queue, to cause the corresponding

data item to be transferred to the memory.

17. (original) A data processing apparatus as claimed in Claim 16, wherein the

predetermined pipeline stage is an issue stage in the second pipeline, and the partner

pipeline stage is an address generation stage in the first pipeline.

18. (previously presented) A data processing apparatus as claimed in Claim 1,

wherein one of the at least one synchronizing queues is a load queue used when the

coprocessor instruction is a load instruction configured to cause data items to be

transferred from memory accessible by the main processor to the coprocessor, the

predetermined pipeline stage is in the first pipeline and is arranged, when processing one

of said load instructions, to cause to be placed in the load queue a token identifying each

data item to be transferred, and the partner pipeline stage is in the second pipeline and is

configured upon receipt of each token from the load queue, to cause the corresponding

data item to be transferred to the coprocessor.

19. (original) A data processing apparatus as claimed in Claim 17, wherein the predetermined pipeline stage is a write back stage in the first pipeline, and the partner pipeline stage is a write back stage in the second pipeline.

20. (previously presented) A data processing apparatus as claimed in Claim 18 wherein one of the at least one synchronizing queues is a store queue used when the coprocessor instruction is a store instruction configured to cause data items to be transferred from the coprocessor to memory accessible by the main processor, the predetermined pipeline stage is in the second pipeline and is arranged, when processing one of said store instructions, to cause to be placed in the store queue a token identifying each data item to be transferred, and the partner pipeline stage is in the first pipeline and is configured upon receipt of each token from the store queue, to cause the corresponding data item to be transferred to the memory, and wherein the load instruction and store instruction may be vectored coprocessor instructions defining multiple data items to be transferred, and the apparatus further comprises flow control logic, associated with at least one of the load queue and the store queue, configured to send a control signal to the predetermined pipeline stage to stop issuance of tokens by the predetermined pipeline stage whilst it is determined that the associated load or store queue may become full.

21. (previously presented) A data processing apparatus as claimed in Claim 20, wherein the flow control logic is provided for the store queue, the flow control logic

being configured to issue the control signal upon receiving an indication from the main processor that the partner pipeline stage cannot accept a data item.

22. (original) A data processing apparatus as claimed in Claim 21, wherein the load queue is a double buffer.

23. Canceled.

24. (previously presented) A data processing apparatus as claimed in Claim 1, wherein the main processor is configured, when it is necessary to flush coprocessor instructions from both the first and the second pipeline, to broadcast a flush signal to the coprocessor identifying the tag relating to the oldest instruction that needs to be flushed, the coprocessor being configured to identify that oldest instruction from the tag and to flush from the second pipeline that oldest instruction and any later instructions within the coprocessor.

25. (previously presented) A data processing apparatus as claimed in Claim 24, wherein one or more of said at least one synchronizing queues are flushed in response to said flush signal, with the tag being used to identify which tokens within the queue are to be flushed.

26. Canceled.

27. (previously presented) A data processing apparatus as claimed in Claim 1, wherein a plurality of said coprocessors are provided, with each synchronizing queue coupling a pipeline stage in the main processor with a pipeline stage in one of the coprocessors.

28. (original) A data processing apparatus as claimed in Claim 1, wherein the data processing apparatus has a synchronous design, such that the tokens are caused to be placed in the queue by the predetermined pipeline stage and are caused to be received from the queue by the partner pipeline stage upon changing edges of a clock cycle.

29. (previously presented) A method of synchronization between pipelines in a data processing apparatus, the data processing apparatus comprising a main processor configured to execute a sequence of instructions and a coprocessor configured to execute coprocessor instructions in said sequence of instructions, the main processor comprising a first pipeline having a first plurality of pipeline stages, and the coprocessor comprising a second pipeline having a second plurality of pipeline stages, and each coprocessor instruction being arranged to be routed through both the first pipeline and the second pipeline, the method comprising the steps of:

(a)   coupling a predetermined pipeline stage in one of the pipelines with a partner pipeline stage in the other of the pipelines via a synchronizing queue including a first-in-first-out (FIFO) buffer having a predetermined plurality of entries;

(b)   placing a token in an entry of the synchronizing queue when the predetermined pipeline stage is processing a coprocessor instruction, the token including a tag which uniquely identifies the coprocessor instruction to which the token relates;

(c)   upon receipt of the token from the synchronizing queue by the partner pipeline stage, processing the coprocessor instruction within the partner pipeline stage;

wherein synchronization of the first and second pipelines between the predetermined pipeline stage and the partner pipeline stage is obtained without passing signals with fixed timing between the pipelines.

30. (previously presented) A method as claimed in Claim 29, wherein a plurality of said synchronizing queues are provided, and said steps (a) to (c) are performed for each synchronizing queue.

31. (previously presented) A method as claimed in Claim 29, wherein one of the at least one synchronizing queues is an instruction queue, the predetermined pipeline stage is in the first pipeline and the partner pipeline stage is in the second pipeline, the method comprising the steps of:

at said step (b), placing a token in the instruction queue identifying a coprocessor instruction; and

at said step (c), upon receipt of the token, beginning processing of the coprocessor instruction identified by the token within the partner pipeline stage.

32. (previously presented) A method as claimed in Claim 29, wherein one of the at least one synchronizing queues is a cancel queue, the predetermined pipeline stage is in the first pipeline and the partner pipeline stage is in the second pipeline, the method comprising the steps of:

at said step (b), placing a token in the cancel queue identifying whether a coprocessor instruction at that predetermined pipeline stage is to be cancelled; and

at said step (c), upon receipt of the token from the cancel queue by the partner pipeline stage, and if the token identifies that the coprocessor instruction is to be cancelled, causing that coprocessor instruction to be cancelled.

33. (previously presented) A method as claimed in Claim 29, wherein one of the at least one synchronizing queues is a finish queue, the predetermined pipeline stage is in the first pipeline and the partner pipeline stage is in the second pipeline, the method comprising the steps of:

at said step (b), placing in the finish queue a token identifying permission for a coprocessor instruction at that predetermined pipeline stage to be retired from the second pipeline; and

at said step (c), upon receipt of the token from the finish queue by the partner pipeline stage, and if the token identifies that the coprocessor instruction is permitted to be retired, causing that coprocessor instruction to be retired.

34. (previously presented) A method as claimed in Claim 29, wherein one of the at least one synchronizing queues is a length queue, the predetermined pipeline stage is in the second pipeline and the partner pipeline stage is in the first pipeline, and the method comprises the steps of:

at said step (b), for a vectored coprocessor instruction, placing in the length queue a token identifying length information for the vectored coprocessor instruction; and

at said step (c), upon receipt of the token from the length queue by the partner pipeline stage, factoring the length information into the further processing of the vectored coprocessor instruction within the first pipeline.

35. (previously presented) A method as claimed in Claim 29, wherein one of the at least one synchronizing queues is an accept queue, the predetermined pipeline stage is in the second pipeline and the partner pipeline stage is in the first pipeline, the method comprising the steps of:

at said step (b), placing in the accept queue a token identifying whether a coprocessor instruction in that predetermined pipeline stage is to be accepted for execution by the coprocessor; and

at said step (c), upon receipt of the token from the accept queue by the partner pipeline stage, and if the token identifies that the coprocessor instruction is not to be accepted, causing that coprocessor instruction to be rejected by the main processor.

36. (previously presented)   A method as claimed Claim 29, wherein one of the at least one synchronizing queues is a store queue used when the coprocessor instruction is a store instruction configured to cause data items to be transferred from the coprocessor to memory accessible by the main processor, the predetermined pipeline stage is in the second pipeline and the partner pipeline stage is in the first pipeline, the method comprising the steps of:

at said step (b), when processing one of said store instructions, placing in the store queue a token identifying each data item to be transferred; and

at said step (c), upon receipt of each token from the store queue by the partner pipeline stage, causing the corresponding data item to be transferred to the memory.

37. (previously presented)   A method as claimed in claim 29, wherein one of the at least one synchronizing queues is a load queue used when the coprocessor instruction is a load instruction configured to cause data items to be transferred from memory accessible by the main processor to the coprocessor, the predetermined pipeline stage is in the first pipeline and the partner pipeline stage is in the second pipeline, the method comprising the steps of:

at said step (b), when processing one of said load instructions, placing in the load queue a token identifying each data item to be transferred; and

at said step (c), upon receipt of each token from the load queue by the partner pipeline stage, causing the corresponding data item to be transferred to the coprocessor.

38. (previously presented) A method as claimed in Claim 37 wherein one of the at least one synchronizing queues is a store queue used when the coprocessor instruction is a store instruction configured to cause data items to be transferred from the coprocessor to memory accessible by the main processor, the predetermined pipeline stage is in the second pipeline and the partner pipeline stage is in the first pipeline, the method comprising the steps of:

at said step (b), when processing one of said store instructions, placing in the store queue a token identifying each data item to be transferred; and

at said step (c), upon receipt of each token from the store queue by the partner pipeline stage, causing the corresponding data item to be transferred to the memory; and

wherein the load instruction and store instruction may be vectored coprocessor instructions defining multiple data items to be transferred, and the method further comprises the step of:

(d) for at least one of the load queue and the store queue, sending a control signal to the predetermined pipeline stage to stop issuance of tokens by the predetermined pipeline stage whilst it is determined that the associated load or store queue may become full.

39. (original) A method as claimed in Claim 38, wherein said step (d) is performed for the store queue, at said step (d) the method comprising the step of issuing the control signal upon receiving an indication from the main processor that the partner pipeline stage cannot accept a data item.

-A14-

40. Canceled.

41. (previously presented) A method as claimed in Claim 29, wherein, when it is necessary to flush coprocessor instructions from both the first and the second pipeline, the method further comprises the steps of:

broadcasting a flush signal from the main processor to the coprocessor identifying the tag relating to the oldest instruction that needs to be flushed;

within the coprocessor, identifying from the tag that oldest instruction and flushing from the second pipeline that oldest instruction and any later instructions within the coprocessor.

42. (previously presented) A method as claimed in Claim 41, further comprising the step of flushing one or more of said at least one synchronizing queues in response to said flush signal, with the tag being used to identify which tokens within the queue are to be flushed.

43. Canceled.

44. (previously presented) A method as claimed in Claim 29, wherein a plurality of said coprocessors are provided, with each synchronizing queue coupling a pipeline stage in the main processor with a pipeline stage in one of the coprocessors.

1346946

45. (original) A method as claimed in Claim 29, wherein the data processing apparatus has a synchronous design, such that the tokens are placed in the queue by the predetermined pipeline stage and are received from the queue by the partner pipeline stage upon changing edges of a clock cycle.

X. EVIDENCE APPENDIX

There is no evidence appendix.

XI. RELATED PROCEEDINGS APPENDIX

There is no related proceedings appendix.

1346946